Hidden Markov Model for Gesture Recognition

Jie Yang, Yangsheng Xu

CMU-RI-TR-94-10

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

May 1994

Contents

List of Figures

Abstract

This report presents a method for developing a gesture-based system using a multi-dimensional hidden Markov model (HMM). Instead of using geometric features, gestures are converted into sequential symbols. HMMs are employed to represent the gestures and their parameters are learned from the training data. Based on "the most likely performance" criterion, the gestures can be recognized through evaluating the trained HMMs. We have developed a prototype system to demonstrate the feasibility of the proposed method. The system achieved 99.78% accuracy for an isolated recognition task with nine gestures. Encouraging results were also obtained from experiments of continuous gesture recognition. The proposed method is applicable to any gesture represented by a multi-dimensional signal, and will be a valuable tool in telerobotics and human computer interfaces.

1 Introduction

A gesture interface is an interface where users specify commands by simple gestures such as drawings and actions. For example, with a gesture interface, a robot may be programmed by showing of teleoperators, and a teleoperated robot may not only know to follow human commands but also the meaning of the commands [1]. To develop such an interface, the key issues are how to sense gesture information and how to recognize the gestures from sensed data. To develop a gesture interface, we need some criteria to evaluate its performance, such as: meaningful gestures; suitable sensors; efficient training algorithms; and accurate, efficient, on-line/real-time recognition.

In general, the technology for capturing gestures is expensive; e.g., a vision system or a data-glove is needed. For this reason some graphical devices, such as a mouse, light pen, joystick, trackball, touch tablet, and thumb-wheel, can be employed to provide a simple input to a gesture recognizer. Other possible devices are a foot controller, knee controller, eye tracker, data nose, and tongue-activated joystick.

The gesture recognition problem consists of pattern representation and decision making. Gestures are usually represented by various features, including templates, global transformations, zones, and geometric features. Templates are the simplest features to compute; they are simply the input data in its raw form. Global transformations, such as rotation, translation, or scaling can be employed to reduce the number of features in templates. Zoning is a simple way of deriving features from a path. Space is divided into a number of zones, and the path is transformed into the sequence of zones which the path traverses. Geometric features of a path, such as its total length, total angle, number of times it crosses itself, etc., can be used to represent the global properties of the path.

Several methods have been used for gesture recognition: template-matching [2], dictionary lookup [3], statistical matching [4], linguistic matching [5], neural network [6], and ad hoc methods. Some of the methods are suitable for only one type of feature representation, while others are more generally applicable. Template-matching systems are easy to train because the prototypes in the systems are simply example templates. However, a large number of prototypes can make the use of the template matching prohibitively expensive. When the features are a sequence of tokens from a small alphabet, lookup techniques are efficient for recognition. The drawback of dictionary lookup is that the system is not robust. Statistical matching methods employ statistics of example feature vectors to derive classifiers. The typical statistics used are average feature vector per class, per-class variance of the individual features, and per-class correlations within features. Some statistical matching methods make assumptions about the distributions of features within a class; the performance of such systems tends to be poor when the assumptions are violated. Other statistical matching methods do not have such assumptions, but they require much training data to estimate the form of the distribution. The linguistic approach tries to apply automata and formal language theory to the pattern recognition problem. The major problem with this approach is the need of supplying a grammar for each pattern class. Neural networks have been successfully applied to solve many pattern recognition problems. Their major advantage is that they are built from a large number of simple elements which learn and are able to

collectively solve complicated and ambiguous problems. Unfortunately, they tend to require a large amount of processing power, especially to train.

Several gesture interfaces have been developed. Coleman [7] built a hand-drawn symbol-based text editor with a touch tablet as the input device. Buxton [8] created a musical score editor with a small amount of gesture input by a mouse. Margaret Minsky [9] implemented a system which uses gestures for selection, movement, and path specification to offer a complete Logo programming environment. Rubine [4] produced a system based on statistical pattern recognition techniques for recognizing signal-path gestures (draw with a mouse or stylus) and multiple-path gestures (consisting of the simultaneous path of multiple fingers). The Glove-talk system [6] employs a Data-Glove to control a speech synthesizer. Teaching-by-showing, teleoperation [10], assembly-plan-from-observation [11] are techniques which use gestures for programming a robot.

This report presents a method for developing a gesture interface using the multi-dimensional hidden Markov model (HMM). HMM is a doubly stochastic model and is appropriate for coping with the stochastic properties in gesture recognition. Instead of using geometric features, gestures are converted into sequential symbols. HMMs are employed to represent the gestures, and their parameters are learned from the training data. Based on the most likely performance criterion, the gestures can be recognized by evaluating the trained HMMs. We have developed a system to demonstrate the proposed method. We defined several digits as gestures and used a mouse as the gesture input device. We then employed HMM to learn and recognize these gestures. The feasibility of the method was demonstrated by the experiments in both isolated and continuous gesture recognition. The proposed method has potential applications in telerobotics and a variety of human machine interfacing problems.

2 Hidden Markov Model

A hidden Markov model is a collection of finite states connected by transitions. Each state is characterized by two sets of probabilities: a transition probability, and either a discrete output probability distribution or continuous output probability density function which, given the state, defines the condition probability of emitting each output symbol from a finite alphabet or a continuous random vector.

An HMM can be defined by:

- {S} – A set of states, including an initial state S_I and a final state S_F
- A – The transition probability matrix, $A = \{a_{ij}\}$, where a_{ij} is the transition probability of taking the transition from state i to state j
- B – The output probability matrix, $B = \{b_j(O_k)\}$ for discrete HMM, $B = \{b_j(\mathrm{x})\}$ for a continuous HMM, where O_k stands for a discrete observation symbol, and x stands for continuous observations of k-dimensional random vectors

In this report, we consider only a discrete HMM. For a discrete HMM, a_{ij} and $b_j(O_k)$ have the following properties:

$$a_{ij} \geq 0, \quad b_j(O_k) \geq 0, \quad \forall i, j, k, \tag{1}$$

$$\sum_j a_{ij} = 1 \quad \forall i, \tag{2}$$

$$\sum_k b_j(O_k) = 1, \quad \forall j. \tag{3}$$

If the initial state is of distribution $\pi = \{\pi_i\}$, an HMM can be written in a compact notation

$$\lambda = (A, B, \pi) \tag{4}$$

to represent the complete parameter set of the model.

For a more detailed reference on theory, computation, and application of HMM, the readers are referred to [13]. In order to simplify the learning process, we consider only the first order HMM, which is based on the following assumptions: (1) Markov assumption: a new state is entered based only on the current state; (2) Output-independent assumption: the output probability distribution function depends only on the state at the particular time regardless of when and how the state is entered.

HMM has been successfully applied to speech recognition [12, 13, 14]. Recently it has been studied in force analysis and task context final segmentation [15, 16]. We have previously proposed to use HMM for modeling and learning human skill [17]. In this report we propose the HMM approach to gesture recognition. In general, the concept of HMM can be used in solving three basic problems: the evaluation problem, the decoding problem, and the learning problem. In the learning problem, we provide model parameters in such a way that the model possesses a high probability of generating the observation for a given model and a set of observations. Therefore, the learning process is to establish gesture models according to the training data. In the evaluation problem we can score the match between a model and an observation sequence, which could be used for isolated gesture recognition. In the decoding problem we can find the best state sequence given an observation sequence, which could be used for continuous gesture recognition.

3 HMM-based Gesture Recognition

The HMM approach to gesture recognition is motivated by the successful application of hidden Markov modeling techniques to speech recognition problems. The similarities between speech and gesture suggest that techniques effective for one problem may be effective for the other as well. First, gestures, like spoken languages, vary according to location, time, and social factors. Second, body movements, like speech sounds, carry certain meanings. Third, regularities in gesture performances while speaking are similar to syntactic rules. Therefore, linguistic methods may be used in gesture recognition. On the other hand, gesture recognition has its own characteristics and problems. To develop a gesture interface, some

criteria are needed to evaluate its performance such as meaningful gestures, suitable sensors, efficient training algorithms, and accurate, efficient, on-line/real-time recognition.

Meaningful gestures may be very complex, containing simultaneous motions of a number of points. However, these complex gestures should be easily specifiable. In general, gestures can be specified either by example or by description. In the former, each application has a training session in which examples of different gestures are collected for training the models. The trained models are the representations of all gestures that the system must recognize. In the latter method of specification, a description of each gesture is written in a gesture description language, which is a formal language in which the syntax of each gesture is specified. Obviously, the example method has more flexibility than the description method. One potential drawback of specification by example is the difficulty in specifying the allowable variation between gestures of a given class. This problem would be avoided if the model parameters were determined by the most likely performance criterion. Because gesture is an expressive motion, it is natural to describe such a motion through a sequential model. Based on these considerations, HMM is appropriate for gesture recognition. A multi-dimensional HMM is able to deal with multi-path gestures which are general cases of gesture recognition. Furthermore, a single path gesture can usually be decomposed into 2D or 3D time series in Cartesian space. That is, a single path gesture $g(x, y, z, t)$ can be decomposed into $X(t)$, $Y(t)$, and $Z(t)$. Moreover, a multi-dimensional HMM provides the possibility of using multiple features to increase the recognition rate.

3.1 Approach

The key idea of HMM-based gesture recognition is to use multi-dimensional HMM representing the defined gestures. The parameters of the model are determined by the training data. The trained models represent the most likely human performance and are used to evaluate new incoming gestures. The HMM-based gesture recognition approach can be described as follows:

1. *Define meaningful gestures* – To communicate with gestures, meaningful gestures must first be specified. For example, a certain vocabulary must be specified for a sign language, and certain editor symbols must be given in advance if the gestures are to be used for editing text files.

2. *Describe each gesture in terms of an HMM* – A multi-dimensional HMM is employed to model each gesture. A gesture is described by a set of N distinct hidden states and r dimensional M distinct observable symbols. An HMM is characterized by a transition matrix A and r discrete output distribution matrices B_i, $i = 1, \ldots, r$. Note that only the structures of A and B are determined in this step and the values of elements in A and B will be estimated in the training process.

3. *Collect training data* – In the HMM-based approach, gestures are specified through the training data. It is essential that the training data be represented in a concise and invariant form. Raw input data are preprocessed before they are used to train

the HMMs. Because of the independence assumption, each dimensional signal can be preprocessed separately. In the prototype system discussed later, the preprocessing involves the short-time Fourier transform and vector quantization techniques.

4. *Train the HMMs through training data* – Training is one of the most important procedures in a HMM-based approach. The model parameters are adjusted in such a way that they can maximize the likelihood $P(O|\lambda)$ for the given training data. No analytic solution to the problem has been found so far. However, the Baum-Welch algorithm can be used to iteratively reestimate model parameters to achieve the local maximum.

5. *Evaluate gestures with the trained model* – The trained model can be used to classify the incoming gestures. The Forward-Backward algorithm or the Viterbi algorithm can be used to classify isolated gestures. The Viterbi algorithm can also be used to decode continuous gestures.

3.2 Feature Extraction and Vector Quantization

One of the most important factors determining the performance of an HMM-based system is the input representation. The input gestures are often represented by certain features. Feature extraction and representation are independent of the gesture recognition system. Several factors such as the existence of a preprocessing algorithm, its necessity, its complexity, and its generality must be considered in determining the extent of preprocessing a gesture. In general, there are two approaches to determining the input representation for a recognition system. The first extensively preprocesses the input data to make "important" features prominent and therefore easy for the recognizer to incorporate them in its processing. Most speech recognition systems adopt this approach. In HMM-based speech recognition [18, 13], the sampled speech is first passed through a filter to get rid of noise and is then blocked into frames. Within each frame, the discrete speech samples are analyzed by a linear predictive coding (LPC) technique. Finally, a set of the LPC coefficients represent the speech signal to be recognized. Because the LPC technique can model the spectrum of the vocal tract as a spectrum of an order-p all-pole model, the method works well in practice.

The second approach just gives the recognizer the "raw" input and allows it to learn from experience which features are important. The successful example of this approach is ALVINN (Autonomous Land Vehicle In a Neural Network) [19]. Because of difficulties with preprocessing images for mobile robot guidance, ALVINN adopts the philosophy of keeping the system general by performing a minimal amount of preprocessing so that the recognizer learns not only how to combine important features for the most appropriate response in the current situation, but also how to detect the most important features in the input image.

The major features for gesture recognition are templates, global transformations, zones, and geometric features. Among these features, templates are the simplest features to compute; they are simply the input data in its raw form. For a path, a template is composed of the coordinates of those points which make up the path. An advantage of templates is that the features are simple to compute. The major disadvantage is that the size of the feature data

increases with the size of the input, making the features unsuitable as input to certain kinds of recognizers. Moreover, template features are not robust within a given class.

Global transformations, such as rotation, translation, or scaling, can be employed to reduce the number of features in templates. The transformations are often chosen in such a way that they are invariant under rotation, translation, or scaling of the input data. For example, the Fourier transform can result in features invariant with respect to rotation of the input pattern [20]. Global transformations generally produce a fixed number of features, which provide greater flexibility in choosing the recognizer. The computation of the transformation may be expensive, and the resulting features are usually difficult to interpret directly.

To extract features by zoning, the space of a path is divided into a number of zones, and the path is transformed into the sequence of zones which the path traverses. It is also possible to encode the direction from where each zone is entered. The major advantages of zoning schemes are their simplicity and efficiency. The disadvantages of zoning methods are similar to those for template features.

Geometric features of a path, such as its total length, total angle, number of times it crosses itself, etc., can be used to represent the global properties of the path. Geometric features are widely used in handwriting recognition [21] and are also used in gesture recognition [4]. Geometric features can carry a variety of useful information such as the total path length. Also, geometric features can be applied to various recognizers which require a fixed number of features, or to recognizers which expect a sequence of features. Geometric features tend to be more complex to compute than the other types of features discussed above.

The short-time Fourier transformation (STFT) is chosen as the preprocessing tool in this research for the following reasons. The Fourier transformation and its inverse establish a one-to-one mapping between the time domain and the frequency domain, and fast Fourier transform (FFT) algorithms can be implemented efficiently. Also, the Fourier transform preserves information from the original signal, and ensures that important features are not lost as a result of the FFT. Furthermore, shifting a waveform within the window changes the real and imaginary parts of the frequency domain in such a manner that the square root of the sum of the squares (the magnitude) remains constant. In fact, if a function is given by $x(t)$ and its Fourier transform is $X(f)$, when $x(t)$ is shifted by a constant time, T, i.e., $x(t-T)$, its Fourier transform is

$$X(f)e^{-j2\pi fT},$$

i.e., time shifting affects phase only; the magnitude remains constant throughout.

Although the Fourier transform does not explicitly show the time localization of frequency components, the time localization can be presented by suitably pre-windowing the signal in the time domain. Accordingly, the short-time Fourier transform of a signal $x(t)$ is defined as [24]

$$STFT_x^{\gamma}(t,f) = \int_{t'}[x(t')\gamma^*(t'-t)]e^{-j2\pi ft'}dt'. \tag{5}$$

The STFT at time t is the Fourier transform of the signal $x(t')$ multiplied by a shifted analysis window $\gamma^*(t'-t)$ centered around t. (All integrals are from $-\infty$ to ∞. The superscript * denotes complex conjugation.) Because multiplication by the relatively short

window $\gamma^*(t'-t)$ effectively suppresses the signal outside a neighborhood around the analysis time point $t = t'$, the STFT is simply a local spectrum of the signal $x(t')$ around the analysis window t.

Since gestures are motion paths, and each dimensional signal is assumed to be stochastically independent, the preprocessing is actually performed on a one-dimensional signal. Although human performance is a nonstationary stochastic process over a long interval, it can be considered stationary over a short time interval. Thus, the STFT should give a good spectral representation of the gesture during that time interval. The windows can be overlapped to prevent loss of information. For each dimensional signal, a Hamming window of a certain width is first used to block the signal into frames. The FFT analysis is then performed for every window. Finally, a set of feature vectors is obtained from the amplitude of the FFT coefficients. These feature vectors can be further processed as discussed below.

Because only a discrete HMM is considered, it is necessary to convert the feature vectors into finite symbols. The independent quantization of each signal value or parameter can be achieved by scalar quantization. In contrast, the joint quantization of a block of parameters can be obtained through vector quantization (VQ). The representation of the VQ codeword in the sample space can be the centroid of the corresponding cell as in conventional vector quantization, or can be computed as the probability density function for the corresponding cell. The latter involves computation of the maximum-likelihood estimates when the observation is incomplete. The VQ techniques have been well used to solve quantization or data compression problems [25]. In an HMM-based approach, VQ can play an important role in converting continuous signals into discrete symbols for discrete HMMs such as those used in speech recognition [18, 13]. This section first reviews the principles of conventional vector quantization and then introduces two standard algorithms used for hidden Markov modeling.

A vector quantizer is completely decided by a codebook, which consists of a set of fixed prototype vectors. VQ reduces data redundancy, and this inevitably causes distortion between the original data and the quantized data. A key problem in the VQ technique is to minimize that distortion. A description of the VQ process includes: (1) the distortion measure, and (2) the generation of a certain number of prototype vectors. Two typical VQ techniques which can minimize the distortion are discussed below.

Let $\mathbf{x} = (x_1, x_2, \ldots, x_d) \in R^d$ be a d-dimensional vector, where $\{x_k,\ 1 \leq k \leq d\ \}$ are real-valued, continuous-amplitude random variables. In the VQ process, the vector $\mathbf{x}$ is mapped onto another real-valued, discrete-amplitude d-dimensional vector $\mathbf{z}$, that is,

$$\mathbf{z} = q(\mathbf{x}), \tag{6}$$

where q(·) is the quantization operator. In general, $\mathbf{z}$ is one of the elements in a finite set of values $\mathbf{Z} = \{\mathbf{z_i},\ 1 \leq i \leq L\ \}$, where $\mathbf{z_i} = (z_{i1}, z_{i2}, \ldots, z_{id})$. The set $\mathbf{Z}$ is referred to as the codebook, L is the size of the codebook, and $\{\mathbf{z_i}\}$ is the set of codewords. The size L of the codebook is also known as the number of levels in the codebook.

The quantizer design process is also known as the *training* process. In this process, the d-dimensional space of the original random vector $\mathbf{x}$ is partitioned into L regions or cells

$\{C_i, 1 \leq i \leq L\}$ and each cell C_i is associated with a vector $\mathbf{z_i}$. The quantizer then assigns the codeword $\mathbf{z_i}$ if $\mathbf{x}$ is in C_i, that is,

$$q(\mathbf{x}) = \mathbf{z_i}, \text{ if } \mathbf{x} \in C_i. \tag{7}$$

Any input vector $\mathbf{x}$ that lies in the cell C_i is quantized as $\mathbf{z_i}$. The shape of each cell can be different, and the positions of the codewords corresponding to the cells are determined by minimizing the average distortion associated with the corresponding cells.

To evaluate the quantization error between $\mathbf{x}$ and $\mathbf{z}$, it is necessary to define a distortion measure $\mathrm{d}(\mathbf{x}, \mathbf{z})$ to measure the quantization quality. The distortion measure between $\mathbf{x}$ and $\mathbf{z}$ is also known as a distance measure. The measure must be tractable and computable in order to be analyzed, and also must be subjectively relevant so that differences in distortion values correlate with the quality of quantization. In general, the weighted mean square error distortion can be used such that distortions contributed by quantizing the different parameters are equal. Unequal weights can be introduced to allow certain components to be more important than others. A popular choice for weights is to use the inverse of the covariance matrix of $\mathbf{z}$, i.e.,

$$d(\mathbf{x}, \mathbf{z}) = (\mathbf{x} - \mathbf{z})' \Sigma^{-1} (\mathbf{x} - \mathbf{z}), \tag{8}$$

where Σ is the covariance matrix of $\mathbf{z}$. The above distortion measure can be simplified to a squared error distortion, i.e.,

$$d(\mathrm{x}, \hat{\mathrm{x}}) = \|\mathrm{x}, \hat{\mathrm{x}}\| = \sum_{i=0}^{d-1} (x_i - z_i)^2. \tag{9}$$

The squared error distortion is used in this dissertation for the VQ. The goal of designing an L-level vector quantizer is to partition a d-dimensional space into L cells and associate with each cell a quantized vector. Optimization methods are usually used to minimize certain distortion measures. One criterion for optimizing the vector quantizer is to let the overall average distortion be minimized over all L-levels of the quantizer. The overall average distortion can be defined by

$$\begin{aligned} D(\mathbf{x}, \mathbf{z}) &= E[d(\mathbf{x}, \mathbf{z})] \\ &= \sum_{i=1}^{L} P(\mathbf{z_i}) E[d(\mathbf{x}, \mathbf{z_i}) | \mathbf{x} \in C_i] \\ &= \sum_{i=1}^{L} \int_{x \in C_i} d(\mathbf{x}, \mathbf{z_i}) f(\mathbf{x}, \mathbf{z_i}) d\mathbf{x} \\ &= \sum_{i=1}^{L} P(\mathbf{z_i}) \int_{x \in C_i} d(\mathbf{x}, \mathbf{z_i}) f(\mathbf{x} | \mathbf{z_i}) d\mathbf{x} \\ &= \sum_{i=1}^{L} D_i, \end{aligned} \tag{10}$$

where $E[\cdot]$ is the expectation, $P(\mathbf{z_i})$ is the discrete probability of the codeword $\mathbf{z_i}$, $f(\mathbf{x}|\mathbf{z_i})$ is the multidimensional probability density function of $\mathbf{x}$ given $\mathbf{z_i}$, D_i is the average distortion in cell C_i. The integral is taken over all components of the vector $\mathbf{x}$.

No analytic solution is known to guarantee global minimization of the average distortion measure for a given set of vectors. However, iterative algorithms, which can guarantee a local minimum, are available and work well in practice.

The LBG algorithm proposed by Linde, Buzo, and Gray [26], will be used in this report. The LBG algorithm iteratively splits the training vectors into 2, 4, ..., 2^m partitions and determines the centroid for each partition. The centroid is refined iteratively by k-means clustering. The LBG algorithm is described as follows:

The LBG Algorithm

1. Initialization: Set L (number of partitions or clusters) = 1. Find the centroid of all the training data.
2. Splitting: Split L into $2L$ partitions. Set $L = 2L$.
3. Classification: Classify the set of training data $\mathbf{x}_k$ into one of the clusters C_i according to the nearest neighbor rule.
4. Codebook Updating: Update the codeword of every cluster by computing the centroid in each cluster.
5. Termination 1: If the decrease in the overall distortion D at each iteration relative to the value D for the previous iteration is below a selected threshold, proceed to Step 6; otherwise go back to Step 3.
6. Termination 2: If L equals the VQ codebook size required, stop; otherwise go back to Step 2.

Step 3 and Step 4 of the LBG algorithm are the same as for the k-means algorithm. Various heuristic methods can be employed in the splitting step to find two vectors that are far apart in each partition.

3.3 Recognition

One of the advantages of the HMM-based approach is that a variety of knowledge sources can be combined into a single HMM. By representing all possible knowledge sources as HMMs, the recognition task becomes a search in an enormous HMM. In the case that no knowledge is added, a recognition model can be simply created by putting all gesture models in parallel, and adding an initial state and a final state. The initial state of the recognition model has a null transition to the initial state of each gesture model; the final state of each gesture model has a null transition to the final state of the recognition model. A null transition is a transition which has a transition probability but does not emit any output symbol, and therefore does not consume any time. An example of a gesture recognition network is shown in Figure 1. Figure 1(a) illustrates an HMM network for isolated gesture recognition without adding knowledge. By adding a null transition from the final state of the recognition model

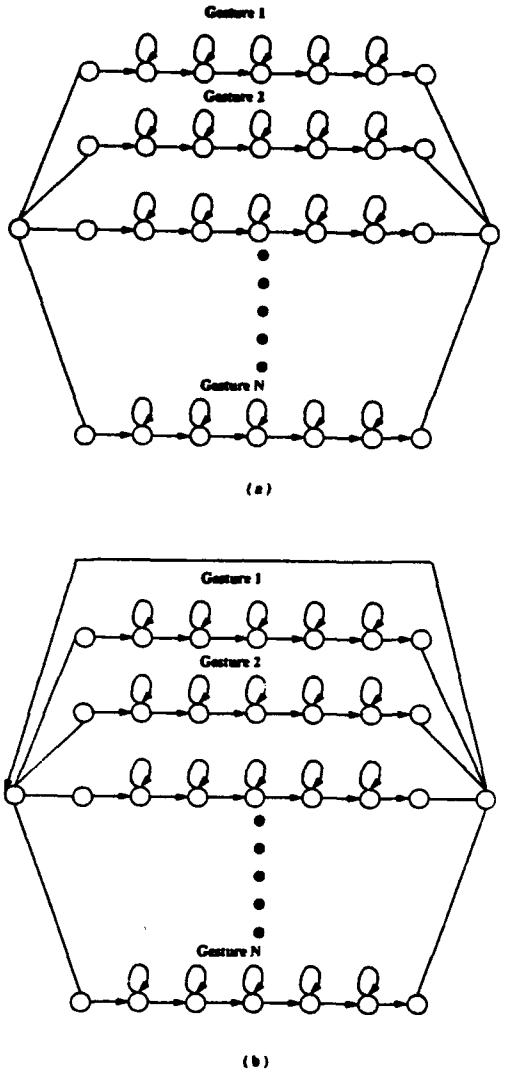

Figure 1: HMM networks for gesture recognition without adding knowledge source.

back to the initial state of the recognition model, a continuous gesture recognition network can be obtained as illustrated in Figure 1(b). The network is more complicated if knowledge sources are encoded into the HMM network.

The goal in a recognition process is to retrieve the input gestures which are represented by a sequence O. The process is to find the HMM with the highest probability given a sequence, i.e.,

$$g^* = \arg \max_{all\ \lambda} P(\lambda|O). \tag{11}$$

The Forward-Backward algorithm is able to evaluate the probability of the observation sequence generated by an HMM, i.e., $P(O|\lambda)$. However, the problem of recognition is to compute $P(\lambda|O)$. From the Bayes formula, a posteriori probability given the sequence can be written as

$$P(\lambda|O) = \frac{P(O|\lambda)P(\lambda)}{P(O)}. \tag{12}$$

Because $P(O)$ is a constant for a given input, only $P(O|\lambda)P(\lambda)$ needs to be computed. $P(\lambda)$ is the probability of the gesture being used, i.e., it characterizes the likely sequence of gestures in certain rules. For example, if the gestures are used for sign language, $P(\lambda)$ can then be determined through a language model. In the simplest case, all the gestures are equally likely to be used, only the term $P(O|\lambda)$ is a variable.

3.4 Isolated vs. Continuous Gesture Recognition

The advantage of hidden Markov modeling is that it can automatically absorb a range of model boundary information for continuous gesture recognition. Other methods, such as neural networks, face problems in training for continuous gesture recognition, because the gesture boundaries are not automatically detectable. Therefore, tedious hand-marking is usually required.

Training HMMs for continuous gesture recognition is similar to that for isolated gesture recognition. To train the parameters of the HMM, the HMMs are concatenated, and each HMM is instantiated with a corresponding gesture. This large concatenated HMM can then be trained by the corresponding data. Because the entire gesture HMM is trained on the entire observation sequence for the corresponding gestures, all possible gesture boundaries are inherently considered. In other words, the parameters of each model will be reestimated based on those states which are suitable for gesture alignment, regardless of the location of the gesture boundaries. Such a training method provides complete freedom to align the concatenated model against the observation, and no effort is required to find gesture boundaries.

The recognition of a continuous gesture is much more difficult than that of an isolated gesture. Because the gesture boundaries cannot be accurately detected, all possible beginning and end points must be considered. This results in a tree search. For a large-vocabulary, continuous gesture recognition task, an optimal full search is infeasible. Several sub-optimal search algorithms can be used instead. The Viterbi algorithm [13] is an efficient sub-optimal search algorithm that can be used for continuous gesture recognition.

4 Computational Consideration

This section discusses several computational considerations in implementing an HMM-based system. These issues are essential for developing a high quality HMM-based system, although some of them are not necessarily required by HMM theory itself.

4.1 Scaling

Scaling is one of the most crucial issues in implementation. Recall that the forward and backward variables, $\alpha(i)$ and $\beta(i)$, consist of the sum of a large number of terms which are

multiplications of the elements of $\{a_{ij}\}$ and $\{b_j\}$. Since each a_{ij} and b_j is less than 1, $\alpha(i)$ and $\beta(i)$ will approach zero in an exponential fashion when the observation length T increases. For the multi-dimensional HMM, there are more multiplications of the output probabilities given the observation length. As a result, it is necessary to use re-scaling techniques. The most straightforward method is to multiply $\alpha(i)$ and $\beta(i)$ by a scaling factor so that they remain within the dynamic range of the computer for $1 \leq t \leq T$. Since the same scalings can be done for $\alpha(i)$ and $\beta(i)$, the scaling factor will be canceled out at the end of the computation.

A scaling coefficient, c_t, can be chosen in such a way that $\sum_i c_t \alpha_t(i) = 1$ for $1 \leq t \leq T$, i.e.,

$$c_t = [\sum_i \alpha_t(i)]^{-1}. \tag{13}$$

The same scaling coefficient c_t can also be applied to $\beta_t(i)$ for $1 \leq t \leq T$ and $1 \leq i \leq N$. In the computation, the forward and backward variables are scaled by c_t at each stage of time t. Because $\alpha(\cdot)$ and $\beta(\cdot)$ are computed recursively in an exponential fashion, the individual scaling factors are multiplied together in the forward and backward recursion. Thus, at time t, $\alpha_t(\cdot)$ is scaled by

$$C_t = \prod_{i=1}^{t} c_i. \tag{14}$$

and $\beta_t(\cdot)$ is scaled by

$$D_t = \prod_{i=t}^{T} c_i. \tag{15}$$

Note that

$$\sum_{i \in S_F} \alpha'_T(i) = C_T \sum_{i \in S_F} \alpha_T(i) \tag{16}$$

$$= C_T P(O|\lambda), \tag{17}$$

where $\alpha'_t(\cdot)$ is scaled $\alpha_t(\cdot)$. Then, the scaled intermediate probability, $\gamma'_t(i,j)$ can be written as

$$\gamma'_t(i,j) = \frac{C_t \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j+1) D_{t+1}}{C_T \sum_{i \in S_F} \alpha_T(i)} \tag{18}$$

$$= \gamma_t(i,j), \tag{19}$$

where $\gamma'_t(\cdot)$ denotes the scaled $\gamma_t(\cdot)$. This implies that the intermediate probabilities can be used in the same way as unscaled probabilities. Therefore, the reestimation formulas can be kept exactly the same as discussed in Chapter II except that $P(O|\lambda)$ should be computed as

$$P(O|\lambda) = \frac{\sum_{i \in S_F} \alpha'_T(i)}{C_T}. \tag{20}$$

In practice, it is not necessary for the scaling operation to be performed at every observation time. It can be applied at any scaling interval where underflow is likely to occur. In the unscaled interval, c_t can be set to unity. An alternative way to avoid underflow is to use a logarithmic representation for all the probabilities, which is discussed below.

4.2 Logarithmic computation

A logarithmic representation offers two advantages in the HMM computation. First, it provides an alternative way to avoid underflow. Underflow cannot happen if all the probabilities are represented in a logarithmic representation. Second, it improves efficiency of the system because integers can be used to represent the logarithmic values, thereby changing floating point to fixed point operations.

When probability P is represented by $\log_b P$, higher precision can be obtained by setting b closer to unity. In the logarithmic representation, multiplication of two numbers implies addition of their logarithms. The addition of two numbers is, however, more complicated. It can, however, be simplified under certain conditions. If $P_1 \geq P_2$, the addition of P_1 and P_2 can be written as

$$\log_b(P_1 + P_2) = \log_b(b^{\log_b P_1} + b^{\log_b P_2}) \quad (21)$$

$$= \log_b P_1 + \log_b(1 + b^{\log_b P_2 - \log_b P_1}). \quad (22)$$

Since logarithms are represented as integers, if $\log_b(1+b^{\log_b P_2 - \log_b P_1})$ is less than 0.5, the sum is equal to $\log_b P_1$. That is, if the magnitude of P_2 is many orders smaller than that of P_1, adding P_1 and P_2 will just result in P_1. An alternative way is to make use of a table. If all possible values of $log_b P_2 - log_b P_1$ are computed in advance, the quantity $\log_b(1+b^{log_b P_2 - log_b P_1})$ can be stored as a table, $T(n)$, where

$$T(n) = \begin{cases} \log_b(1 + b^n) & \text{if } T(n) \geq 0.5 \\ 0 & \text{otherwise.} \end{cases} \quad (23)$$

The value of b is crucial in making such a table. To guarantee computational accuracy, the size of $T(n)$ can be determined by decreasing n from 0 until $\log_b(1 + b^n)$ approaches zero. Therefore, the range of values for n depends upon the value of b. For example, varying b from 1.0001 to 1.00001, the size of $T(n)$ increases from 99,041 to 1,220,614 when 32-bit integers are used for the logarithms.

Using the table $T(n)$, the addition of two probabilities can be implemented as one integer addition, one subtraction, two comparisons, and one table lookup, i.e.,

$$\log_b(P_1 + P_2) = \begin{cases} \log_b P_1 + T(\log_b P_2 - \log_b P_1) & \text{if } P_1 > P_2 \\ \log_b P_2 + T(\log_b P_1 - \log_b P_2) & \text{otherwise.} \end{cases} \quad (24)$$

The errors introduced by the table are of the same order as when the scaling procedure is used in the floating point representation [27, 18].

4.3 Thresholding

The efficiency of computation in the Baum-Welch reestimation formulas can be improved by thresholding the forward and backward variables. In the parameter reestimation procedure, the final reestimations depend on the summation of γ's. If a certain γ is very small relative to other γ's, it makes little contribution to the final reestimations. The forward variable $\alpha_t(i)$ and the backward variable $\beta_t(i)$ are major factors in γ. Therefore, if a certain $\alpha()$ or $\beta()$ become very small relative to other α's or β's during the computation, they can be set to zero without significantly affecting the performance. A method to establish such a threshold is introduced below [13].

Define $\overline{\alpha}_t$ as the maximum $\alpha_t(i)$ at time t with respect to different states i,

$$\overline{\alpha}_t = \max_i \alpha_t(i). \tag{25}$$

Given a threshold c, in each state i if $\alpha_t(i) < c\overline{\alpha}_t$, set $\alpha_t(i)$ equal to zero before moving on to compute α at time $t+1$. Because at time $t+1$, only those α from time t which are greater than zero will be included, this thresholding is very useful in reducing computation to a manageable size. The backward variable can be thresholded in the same way. Moreover, if $\alpha_t(i)$ is zero, $\beta_t(i)$ can also be set to zero.

The selection of the threshold c is crucial in this method. If c is too small, more than the necessary computation is performed. If c is too large, the forward-backward algorithm will deteriorate to the Viterbi training, i.e., only the best path is used for the estimation of model parameters. There is no analytic solution for c and the appropriate value of c can only be determined empirically.

4.4 Multiple Independent Sequences

The parameter estimation algorithms described in Chapter II are for only one training datum, but are easily generalized in our application where multiple training sequences are needed. Note that the Baum-Welch algorithm does nothing more than compute the frequencies of occurrence of various events. The modification for multiple independent sequences requires computing the frequencies of occurrence in each sequence separately and adding them together [13].

4.5 Initialization

Theoretically, the Baum-Welch algorithm gives only a local maximum of the likelihood function. If a poor initialization point is given, a poor local maximum may be reached. Particularly, if a probability is initialized zero, it will remain zero with every iteration. In practice, finding the unique global maximum seldom matters. What does matter is finding a set of parameters that recognizes the gestures with high accuracy. Fortunately, our experience

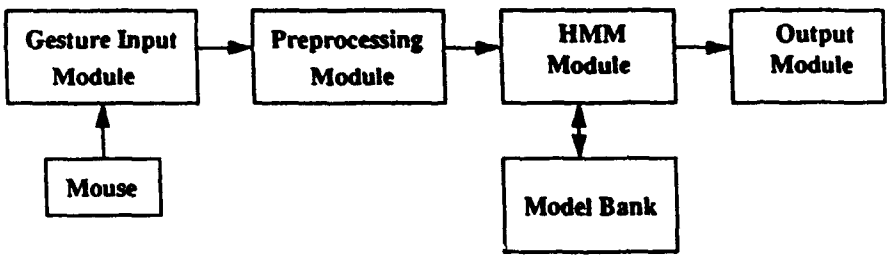

Figure 2: Block diagram of the gesture based system.

shows that, for a discrete HMM, uniformly distributed initial points work well. A similar observation is also given by other researchers [13]. Another advantage of using discrete HMMs is that the output distributions are automatically learned by the training process. For the continuous or semi-continuous HMM, however, good initial values are essential for obtaining a good model.

5 Experimental Results

In order to demonstrate the proposed method, a prototype system has been developed (Figure 2). The system was designed for handdrawn gesture recognition. The gesture trace is produced using a mouse and the system is interfaced with a X window. The input gestures are resampled and then converted into symbols. Each gesture is described by a two-dimensional HMM. The parameters of the HMMs are learned from the training data.

The system was run successfully for some case studies. Nine gestures were defined (Figure 3), and a six-state, two-dimensional Bakis model was employed to model each gesture. For $n = 6$, the form of the transition matrix A, the initial state probabilities, and the state transition coefficients of state 6 were obtained from [17] Two 256×6 observability matrices were used, with each column representing the observation probability distribution for one state. The learning algorithms can be obtained from [17].

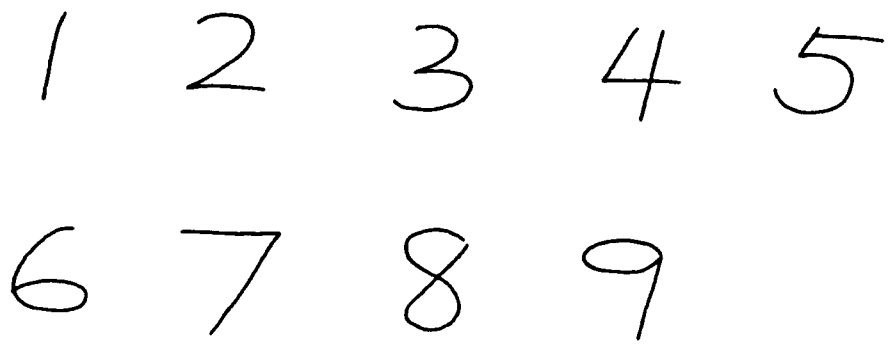

Figure 3: The defined gestures.

5.1 Gesture Input

A mouse is a two-dimensional single-path gesture input device. Each gesture can be represented as an array g of P time-sampled data points:

$$g_p = (x_p, y_p, t_p) \quad 0 \leq p < P.$$

The sampling time information t_p is needed because the X-window interface does not deliver input points at regular intervals. The two-dimensional single-path gesture g_p can be projected onto the x and y axes, and two independent one-dimensional signals, $x(t_p)$ and $y(t_p)$, are obtained. The $x(t_p)$ and $y(t_p)$ are then re-sampled by linear interpolation to obtain input points with the same sampling interval. Interpolation is the same operation as "table lookup." Described in "table lookup" terms, the "table" is $[T, X]$. Linear interpolation "looks-up" the elements of t_i in T, and, based upon their location, returns values x_i interpolated within the elements of X.

5.2 Preprocessing

The experiments were run on a SUN4 machine. The gestures were generated with a mouse at about 40 millisecond sampling time intervals and then were resampled at 20 millisecond sampling time intervals. The FFT and VQ techniques were used for pre-processing the gestures. The Hamming window was first used with a width of 320 ms in every 160 ms. FFT analysis was then performed for every window. Finally, a set of 16-dimensional vectors was obtained from the amplitude of the FFT coefficients. The LBG algorithm was used to produce the VQ codebooks. First, a certain number of the training vectors was used to generate a 256-vector codebook for each dimensional signal. These sets of 256 vectors were the symbols in the output probability distribution function in our discrete HMM. An input

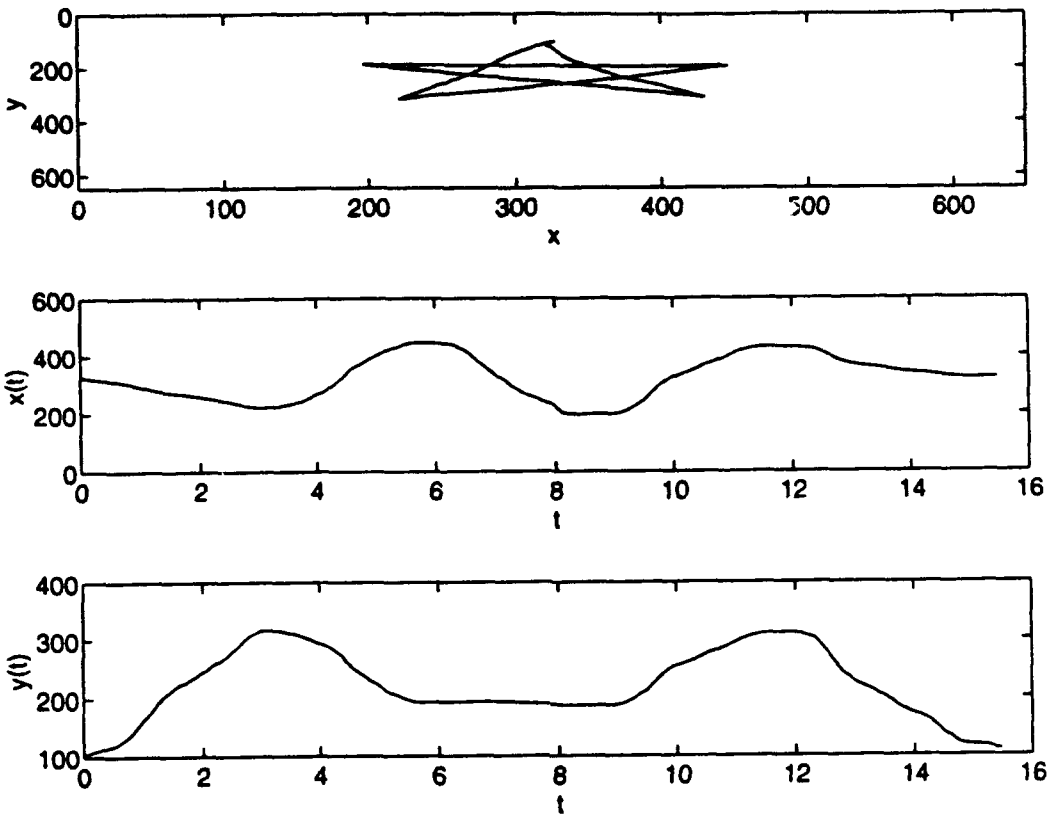

Figure 4: An example of gesture signals.

vector corresponded to the set of 16 32-bit floating point FFT coefficient amplitudes, and was mapped onto an 8-bit index which represents one of the 256 prototype vectors. Figure (4) shows an input gesture can be decomposed into two one-dimensional signals, and Figure (5) illustrates an example of preprocessing for a one-dimensional signal.

5.3 Training

To initialize the model parameters, output probabilities were set equal to $\frac{1}{256}$, where 256 is the VQ level. The transition probabilities were initialized with uniformly distributed random number. With these initial parameters, the Forward-Backward algorithm was run recursively on the training data. The Baum-Welch algorithm was used iteratively to reestimate the parameters based on the forward and backward variables. After each iteration, the output probability distributions were smoothed using a floor between 0.0001 and 0.00001, and renormalized to meet stochastic constraints.

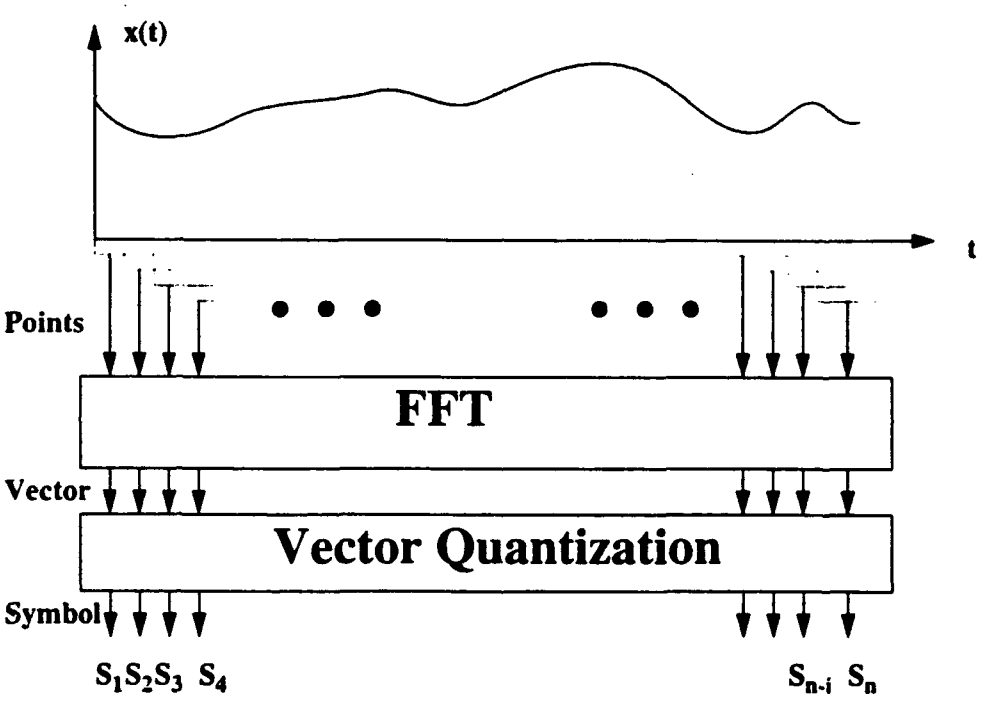

Figure 5: An example of preprocessing for one-dimensional signal.

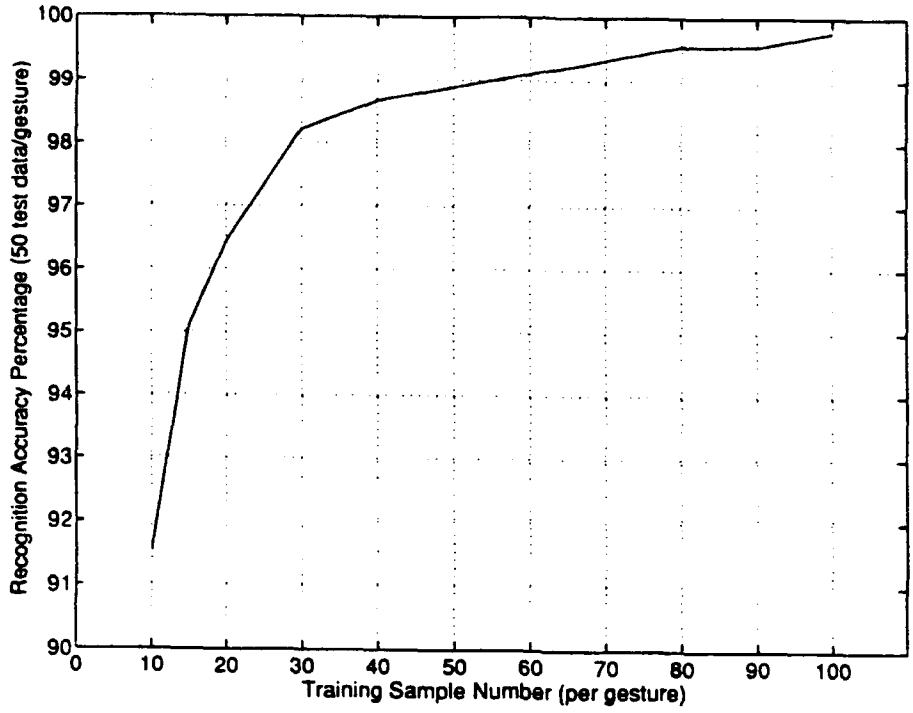

Figure 6: Recognition rate vs. training set size.

5.4 Results

First, isolated gesture recognition was investigated. One hundred fifty samples of data were collected for each gesture, the first 100 for training and the rest of the samples for testing. For a total of 450 testing samples, the system successfully recognized 449 samples, a 99.78% accuracy rate. The effect of the training set size was studied by varying the number of training samples per gesture. Figure 6 shows the result of the recognition rate versus training set size. The testing set size was fixed at 50 per gesture (total 450) while the training set size changed from 10 to 100 samples. As expected, the recognition rate increased as the training set size increased. When the training set size was 10 samples per gesture, the recognition rate was 91.56%, and when the training set size increased to 100 samples per gesture, the rate was 99.78%.

The performance of the HMM-based system for continuous gesture recognition was then studied. The focus was on the capability of the system to separate connected gestures. Continuous gestures may be either connected or separated from each other, as shown in Figure 7. The gesture models of "2" and "8" were trained together with both connected and separated "82"s. Then the trained models were tested by inputing gesture "2", "8", a connected "82", and a separated "82". In all cases, the system demonstrated the capability to recognize the gestures correctly.

From the experimental results, some remarks are in order:

(a) (b)

Figure 7: An example of continuous gesture: (a) connected and (b) separated.

1. We have demonstrated that HMM is a feasible parametric model for building a gesture-based system. Although we have examined only a two-dimensional single-path gesture case in this report, the method can be extended to multiple-path applications. For example, Data-Glove is a typical multiple-path gesture input device that allows us to encode the gestures in the joint space and represent the gestures by multi-dimensional HMM. The same method is applicable to develop a variety of gesture recognition systems that can be useful in telerobotics and human machine interfacing.

2. HMM is a doubly stochastic model and is appropriate for coping with the stochastic properties in gesture recognition. HMM can represent all the training data in the statistic sense by its parameters and its parameters can be optimized by efficient algorithms for the accurate estimation. Therefore, the model can be updated incrementally, which is desirable for *learning*. Comparing with neural network approach, HMM converges faster because of its efficient reestimation algorithms, and is more suitable for continuous gesture recognition because no hand-marking is needed.

3. The gesture recognition of a two-dimensional single-path has much in common with on-line handwriting recognition. However, our method can potentially deal with the gestures which dimensions are other than two, are drawn from unusual symbols, specify entire commands, vary in size and orientation, and have a dynamic components.

4. Handwriting recognition systems can be broadly grouped into two classes: on-line and off-line. In on-line handwriting recognition, characters are recognized as they are drawn. In off-line handwriting recognition, characters are first drawn on paper, and then optically scanned and represented as two-dimensional rasters. Although gesture recognition is different from handwriting recognition in general, it is possible to apply the proposed method to handwriting recognition after modification. For example, the method is able to do on-line handwriting recognition by taking the FFT base on

arc length instead of time t. The same idea can be applied to off-line handwriting recognition after some preprocessing.

6 Conclusion

We have proposed a method for modeling, recognizing, and learning human gestures using the hidden Markov model. HMM is a powerful parametric model and is feasible to characterize two stochastic processes – the measurable action process and immeasurable, hidden mental states. Instead of using geometric features, we convert the gestures into sequential symbols. HMMs are employed to represent the gestures, and their parameters are learned from the training data. Based on the most likely performance criterion, the gestures can be recognized by evaluating the trained HMMs.

We developed a prototype to demonstrate the feasibility of the HMM-based gesture recognition method. We defined several digits as gestures and used a mouse as the gesture input device. We then applied HMM to learn and to recognize measured gestures. The experimental results showed that the proposed method can be used for learning and recognizing gestures in both isolated cases and continuous cases. The proposed method is applicable to multi-dimensional signal recognition and learning problems, and will be of significance in developing gesture interface in telerobotics and human-computer interfacing.

Acknowledgement

The authors would like to thank Dr. X.D. Huang, Dr. T. Kanade, and Dr. C.S. Chen for their valuable suggestions and discussions.

References

[1] T.H. Speeter, " Transformation human hand motion for telemanipulation," *Presence*, Vol. 1, No.1, pp.63-79, 1992.

[2] J. S. Lipscomb, "A trainable gesture recognizer," *Pattern Recognition*, vol. 24, No. 9, pp895-907, 1991.

[3] W. M. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics*, McGraw-Hill, 1979.

[4] D. H. Rubine, "The automatic recognition of gesture," *Ph.D dissertation*, Computer Science Department, Carnegie Mellon University, December, 1991.

[5] K. S. Fu, "Syntactic recognition in character recognition," *Volume 112 of Mathematics in Science and Engineering*, Academic Press, 1974.

[6] S. S. Fels and G. E. Hinton, "Glove-talk: a neural network interface between a data-glove and a speech synthesizer," *IEEE Trans. Neural Networks*, Vol. 4, No. 1, pp.2-8, 1993.

[7] Michael L. Coleman, "Text editing on a graphic display device using hand-drawn proof-reader's symbols," *Proceedings of the Second University of Illinois Conference on Computer Graphics*, University of Illinois Press, pp. 283-290, 1969.

[8] W.A.S. Buxton, R. Sniderman, W. Reeves, S. Patel, and R. Baecker, "The evolution of the SSSP score-editing tools," *Foundation of Computer Music*, Chapter 24, pp.443-466, MIT press, 1985.

[9] M.R. Minsky, "Manipulating simulated objects with real-world gestures using a force and position screen," *Computer Graphics*, Vol. 18, No. 3, pp. 195-203, 1984.

[10] S. Hirai and T. Sato, "Motion understanding for world model management of telerobot," *Proceedings of IROS'89*, pp. 124-131, 1989.

[11] K. Ikeuchi and T. Seuhiro, "Towards man assembly plan from observation: task recognition with polyhedral objects," CMU Tech. Rep. CMU-CS-91-167, Computer Science Department, Carnegie Mellon University, 1991.

[12] K.F. Lee, H.W. Hon and R. Reddy, "An overview of the SPHINX speech recognition system," *IEEE Trans. on ASSP*, Vol. 38, No. 1, pp. 35-45, 1990.

[13] X.D. Huang, Y. Ariki and M. A. Jack, "Hidden Markov Models for Speech Recognition," *Edinburgh University Press*, 1990.

[14] X.D. Huang, "Phoneme classification using semicontinuous hidden Markov models," *IEEE Trans. on ASSP*, Vol. 40, No. 5, pp. 1062-1067, 1992.

[15] B. Hannaford, P. Lee, "Hidden Markov model analysis of force/torque information in telemanipulation," *The International Journal of Robotics Research*, Vol. 10, No. 5, pp.528-539, 1991.

[16] P. K. Pook and D. Ballard, "Recognizing teleoperated manipulations," *Proceedings of 1993 IEEE Inter. Conf. on Robotics and Automation*, Atlanta, Georgia, USA, Vol. 2, pp.578-585, 1993.

[17] J. Yang, Y. Xu and C. S. Chen, "Hidden Markov model approach to skill learning and its application in telerobotics," *Proceedings of 1993 IEEE Inter. Conf. on Robotics and Automation*, Vol. 1, pp.396-402, 1993.

[18] K.F. Lee, "Large-vocabulary speaker-independent continuous speech recognition: The SPHINX system," Ph.D. thesis, Department of Computer Science, Carnegie Mellon University, 1988.

[19] D.A. Pomerleau, "Neural Network Perception for Mobile Robot Guidance," Ph.D. dissertation, Department of Computer Science, Carnegie Mellon University, 1992.

[20] G.H. Granlund, "Fourier preprocessing for hand print character recognition," *IEEE Trans. on Computers*, Vol. 21, pp. 195-201, 1972.

[21] C.Y. Suen, M. Berthod, and S. Mori, "Automatic recognition of handprinted characters: the state of the art," *Proceedings of the IEEE*, Vol. 68, No. 4, pp. 469-487, 1980.

[22] L.E. Baum, T. Petrie, G. Soules, and N. Weiss, " A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chins," *Ann. Math. Stat.*, Vol. 41, No. 1, pp. 164-171, 1970.

[23] R.W. Schafer and L.R. Rabiner, "Digital representations of speech signals," *Proceedings of IEEE*, Vol. 63, No. 4, pp. 662-677.

[24] F. Hlawatsch and G.F. Boudreaux-bartels, "Linear and quadratic time-frequency signal representations," *IEEE SP Magazine*, Vol.9, No.2, 1992.

[25] R.M. Gray, "Vector quantization," *IEEE ASSP Magazine*, Vol. 1, No.2, pp. 4-29, 1984.

[26] Y. Linde, A. Buzo, and R.M. Gray, " An algorithm for vector quantizer design," *IEEE Trans. on Communication*, Vol. COM-28, pp.84-95, 1980.

[27] Peter F. Brown, "The Acoustic-Modeling Problem in Automatic Speech Recognition," Ph.D. thesis, Department of Computer Science, Carnegie Mellon University, 1987.